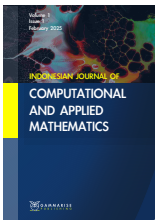
Indonesian Journal of Computational and Applied Mathematics
https://ejournal.gammarisepub.com/index.php/indocam
Volume 1, Issue 2, pages 76–88, June 2025
https://doi.org/10.64182/indocam.v1i2.20, E-ISSN 3090-5281



Research Article

Developing a *Python*-Based Application for a Discrete-Time Population Dynamics Model

Farhah Nadhilah[1], **Hasan S. Panigoro**[2,✉], **Armayani Arsal**[1], **Nurwan**[1], **Djihad Wungguli**[1], **Isran K. Hasan**[3]

[1]Department of Mathematics, Universitas Negeri Gorontalo, Bone Bolango 96554, Indonesia
[2]Biomathematics Research Group, Universitas Negeri Gorontalo, Bone Bolango 96554, Indonesia
[3]Department of Statistics, Universitas Negeri Gorontalo, Bone Bolango 96554, Indonesia

Abstract

Difference equation is a type of equation in mathematics that is widely used to describe certain phenomena as time changes, one of which is in the field of population dynamics. In various studies, it is explained that solving complex population dynamics models is by using numerical simulations. Along with the development of technology, computational science is used to help solve mathematical problems that are difficult to solve analytically. One of them is to use a programming language, such as Python, to help present data in a graphical form. This research aims to develop an application that presents a computational solution to a difference equation using Python. The numerical results begin by entering the equation and variable values into the application, which then automatically generates a figure according to the entered equation. The figures generated in the application include one-dimensional and two-dimensional time series, as well as a Bifurcation diagram.

Keywords : *Difference equation · Python · Numerical simulation · Time series · Bifurcation diagram*
MSC2020 : *65L05 · 92D25 · 37N25 · 65Q10 · 00A69*

1. Introduction

Difference equations have a long history of being used to model population dynamics using autonomous discrete time [1]. Until now, the use of difference equations has been widely used to solve mathematical models. Research that uses difference equations to solve mathematical models is research conducted by [2–5], which utilizes difference equations to solve mathematical models in the field of population dynamics. Recently, one of the supporting things used to solve population dynamics models is the ability to present solutions numerically. The numerical method is a technique used to solve many mathematical problems that are difficult to solve using analytical methods [6]. In various studies, it is explained that solving complicated population dynamics is by performing numerical simulations [7–10]. Guedes et al. [11] provides an explanation that, for a better understanding of computational analysis, it is important to discretize the model into discrete time. The use of the Forward Euler Method is one of the supports used to facilitate numerical solutions to discrete models [12].

The presentation of solutions using graphs as a result of numerical simulations is very helpful for researchers to see the long-term solution of the resulting model. There are two types of graphical plots often used by researchers to present models: time series and bifurcation diagrams. Time Series graphs in population dynamics serve to visualize changes in variables over time. Using Time Series graphs, predictions can be made about future systems based on documented time patterns [13]. Bifurcation

✉ Corresponding author. Email: hspanigoro@ung.ac.id
received: 26 June 2025, revised: 13 August 2025, accepted: 4 September 2025, published Online: 6 October 2025

graphs are often used to identify specific elements in a system that can influence the resulting time series response and overall dynamics [14]. By analyzing bifurcation diagrams, it is possible to identify parameter ranges that will lead to various types of conditions, such as stable equilibrium points, cycle limits, or chaos dynamics [15].

The presentation of solutions in the form of graphs is certainly inseparable from the help of computational science. Computational science is a field that provides solutions to complex mathematical problems that are difficult to solve analytically, instead solving them numerically. This computing resource refers to a computer [16]. *Python* is one of the programming languages that can be used to perform numerical simulations. *Python* is a dynamic, high-level programming language that can convert source code into machine code directly when the program is run. *Python* supports object-oriented programming, procedural programming approaches, and provides dynamic memory allocation [17, 18]. *Python* is a very good language used by beginners with simple and clear syntax [19]. In addition to syntax that is easy to understand, there are other advantages of the *Python* programming language, including having many libraries that can be used to analyze data, create graphs, and create GUIs such as *NumPy*, *Matplotlib*, and *TkInter* [20]. In making a GUI (Graphic User Interface), several packages from the *Python* library are needed, including *Tkinter* and *Matplotlib*.

Tkinter is a module designed for building graphical user interfaces (GUIs), while *Matplotlib* focuses on creating graphs and data visualizations. Tkinter serves as *Python's* interface to the Tk GUI toolkit and has been included in *Python's* standard library since the release of version 1.1 in 1994 [21]. *Matplotlib* is built on the principle of allowing users to create simple plots with only a few commands, or even a single command. The structure of *Matplotlib* is divided into three main components: first, the pylab interface, which consists of functions from matplotlib.pylab that enable users to create plots with MATLAB-like syntax; second, the *Matplotlib* frontend or API, a set of classes that handle the creation and management of images, text, lines, and plots; and third, the backend, a drawing tool that uses a renderer to translate the frontend's representation into either a display or printed output [22].

In this article, we provide a new *Python*-based application for discrete time dynamics models that has the following main contributions it can graph time series and bifurcations for difference equations constructed using forward Euler. To describe the results, we organize the article as follows: Section 2 explains the method of creating applications using the waterfall method. In Section 3, we demonstrate the successful simulation of time series graphs and bifurcations, and validate the application using the equations presented in the reference article. The last, in Section 4, are the conclusions of this research.

2. Methods

This research method begins with conducting a literature study. This research studies books, journals, scientific articles, other literature sourced from the internet regarding *Python*-based numerical simulations for two-dimensional discrete dynamical systems in the field of population dynamics with the aim of obtaining all the information needed in this research. Furthermore, the Waterfall method is used as a software development method that is often used. This development method approaches systematically and sequentially, using the waterfall method the next stage will not be implemented before the previous stage is completed [23].

This type of research is qualitative research and development research. Qualitative research is conducted using data based on word groupings and not the number. Then the development research carried out by producing an application made using the *Python* programming language.

3. Results and Discussion

The result of this research is an application that can be used to simulate discrete dynamic system graphs in the form of Time Series and Bifurcation graphs with the waterfall method which is divided into five parts, below will explain the process of making applications using the waterfall method.

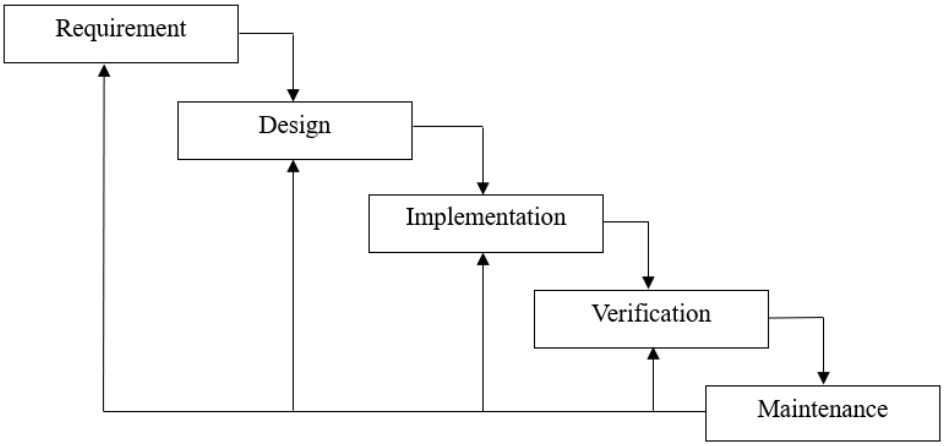


Figure 1. Waterfall Diagram

3.1. System Analysis and Design Based on Waterfall Method

3.1. Requirement

The analysis of the needs needed in this application includes functional requirements that can function to produce one-dimensional and two-dimensional Time Series and Bifurcation graphs by entering equations and parameters manually by the user and then the graph can be saved on the user's device.

3.1. Design

In the design section there are twelve different windows that each have their own function. In the initial display there is a button for PRATINJAU APLIKASI which serves to explain the functional items that will be used in making graphics, then there is a START button which serves to start the application, the TENTANG APLIKASI button serves to provide information about the application developer and the purpose of the application created.

In the PRATINJAU APLIKASI page, there are four different sections for one-dimensional and two-dimensional Time Series graphs and one-dimensional and two-dimensional Bifurcation graphs, which explain the function of each item in the simulation.

In this page, there are four different sections, each of which contains item descriptions for each window used in charting

Then the TENTANG APLIKASI item will be displayed below. This document explains the purpose of the application and includes information about the application designer.

After pressing the START button, the user will enter the display to start the simulation, there are five different button options divided into two sections. The top section has buttons for one-dimensional Time Series and Bifurcation. The bottom section has buttons for Time Series and two-dimensional bifurcation. The last button is BACK to return to the home page.

The Two-Dimensional Time Series page has several sections in it that are almost similar to the One-Dimensional Time Series. There are two different EQUATION to enter two different equations, for each equation there are x and y variables. ITERATION section to fill the iteration value desired by the user, STEP SIZE to fill the step value from the starting point to the next point, INVAL (INITIAL VALUE) 1 and INVAL (INITIAL VALUE) 2 to enter the initial value of each equation, X1_LABEL and Y1_LABEL, to name the x and y labels of the first graph, X2_LABEL and Y2_LABEL to name the x and y labels of the second graph, PLOT button to plot the graph,

Figure 2. First page (Home View) of the application

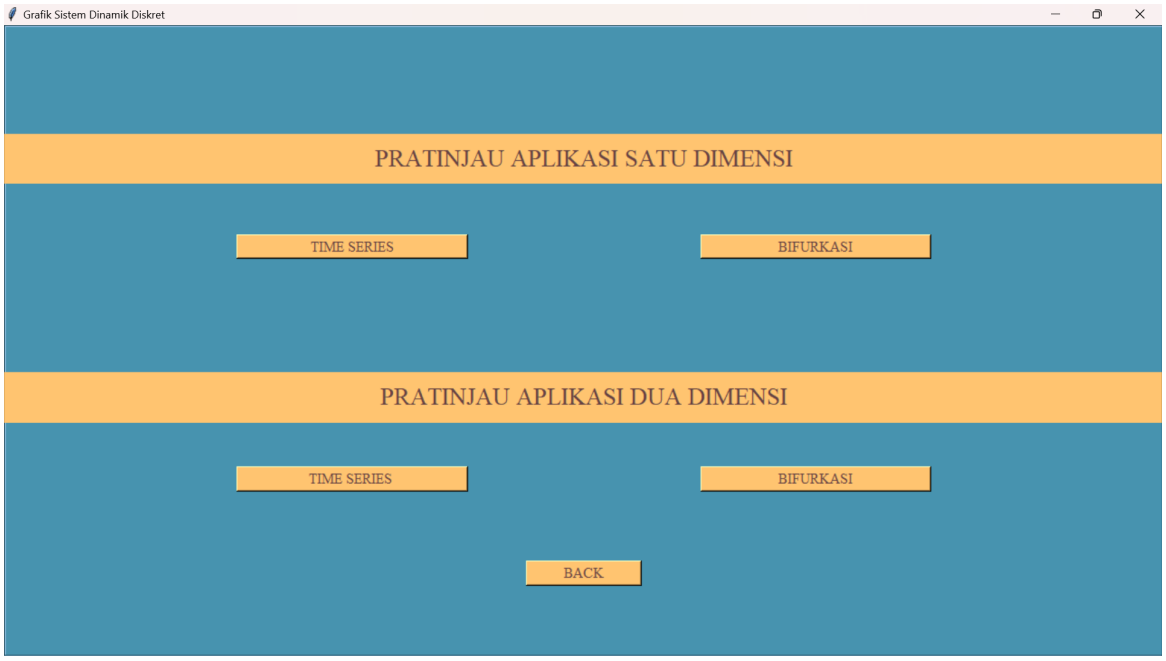


Figure 3. The application preview page

SAVE PLOT button to save the graph that has been plotted, and BACK button to return to the second page of the application.

The Bifurcation page for two dimensions is not much different from the one-dimensional Bifurcation page, there are several sections in it. Two different EQUATION to enter different equations that will be plotted, ITERATION to enter the iteration value desired by the user, TAIL POINT to determine the final value, INVAL (INITIAL VALUE) 1 and INVAL (INITIAL VALUE) 2 to enter the initial value of each equation, enter the value of $\Delta 0$, ΔN, and NUM POINT, X1_LABEL and Y1_LABEL to name the x and y labels of the first graph, X2_LABEL and Y2_LABEL to name the

Figure 4. The application preview page

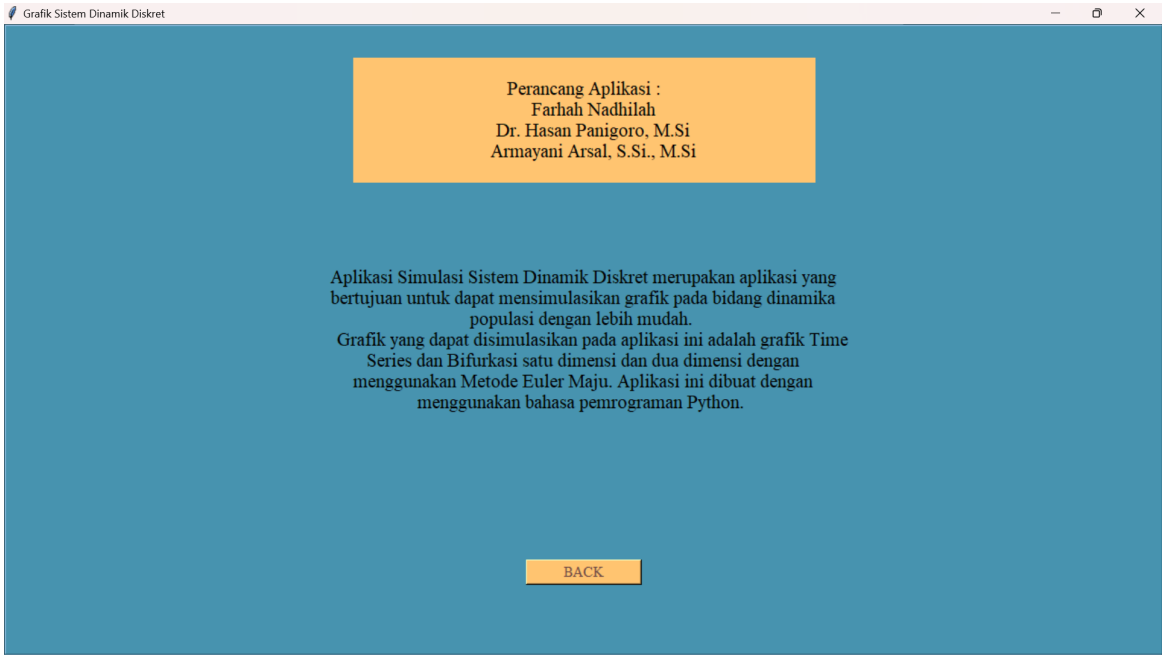


Figure 5. The "About Application" page

x and y labels of the second graph, PLOT to plot the graph, SAVE PLOT button to save the plotted graph, BACK button to return to the second page of the application.

3.1. Implementation

The implementation that will result from this research is a discrete dynamic system simulation application to produce one-dimensional and two-dimensional graphs made using Spyder software using the *Python* programming language. This program enables users to easily produce discrete dynamic system graphs by entering equations and variable values within the application. Below, the

Figure 6. The "Start" page

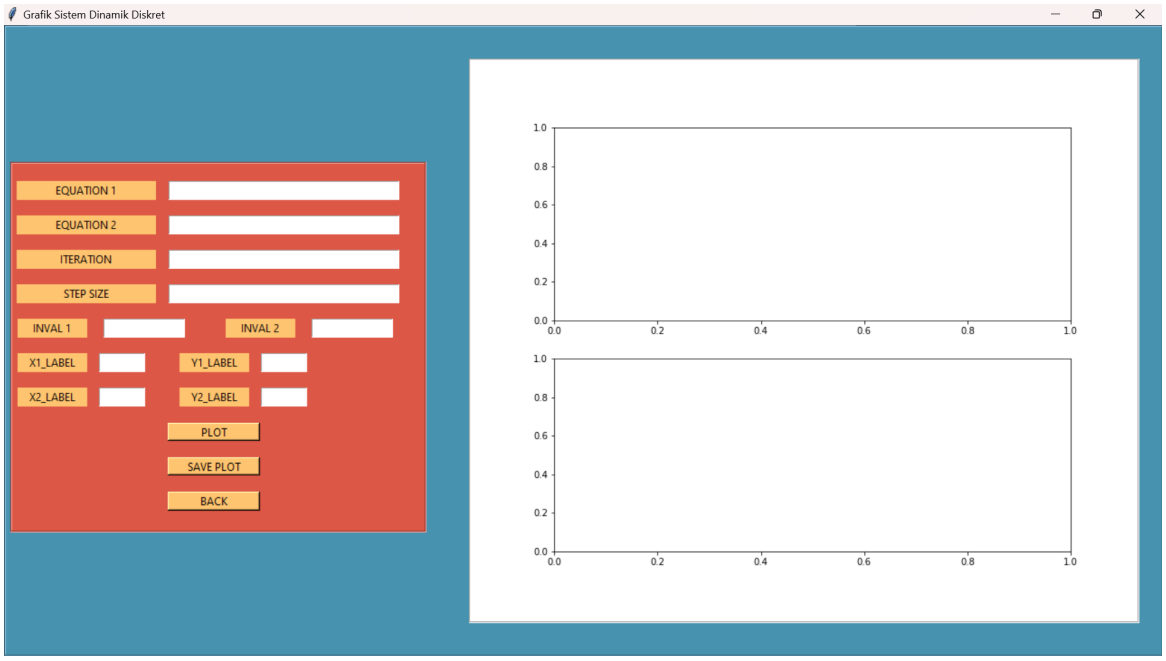


Figure 7. Two-dimensional time-series page

Euler Forward Discrete Dynamical System Simulation application will be explained in more detail. This section is the implementation stage of the program, which involves creating two-dimensional discrete dynamic system graphs on Time Series graphs. It begins by filling in the EQUATION 1 and EQUATION 2 sections with the corresponding equations. The equation in this process only involves the variables x and y for other parameters can be written directly, enter the ITERATION value which serves to model the variable changing over time, enter the STEP SIZE value which serves as the time distance or interval between the starting point to the next point in the curve, enter the initial value for INVAL 1 used for EQUATION 1, as well as for INVAL 2 used for EQUATION 2, then enter

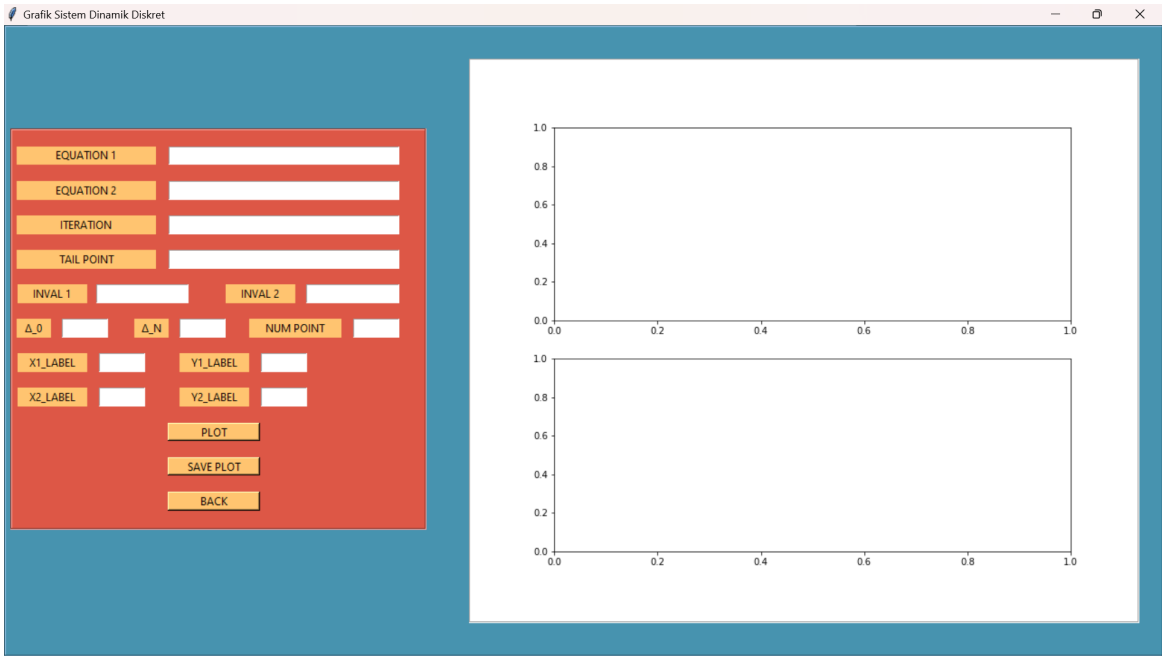


Figure 8. Two-dimensional bifurcation page

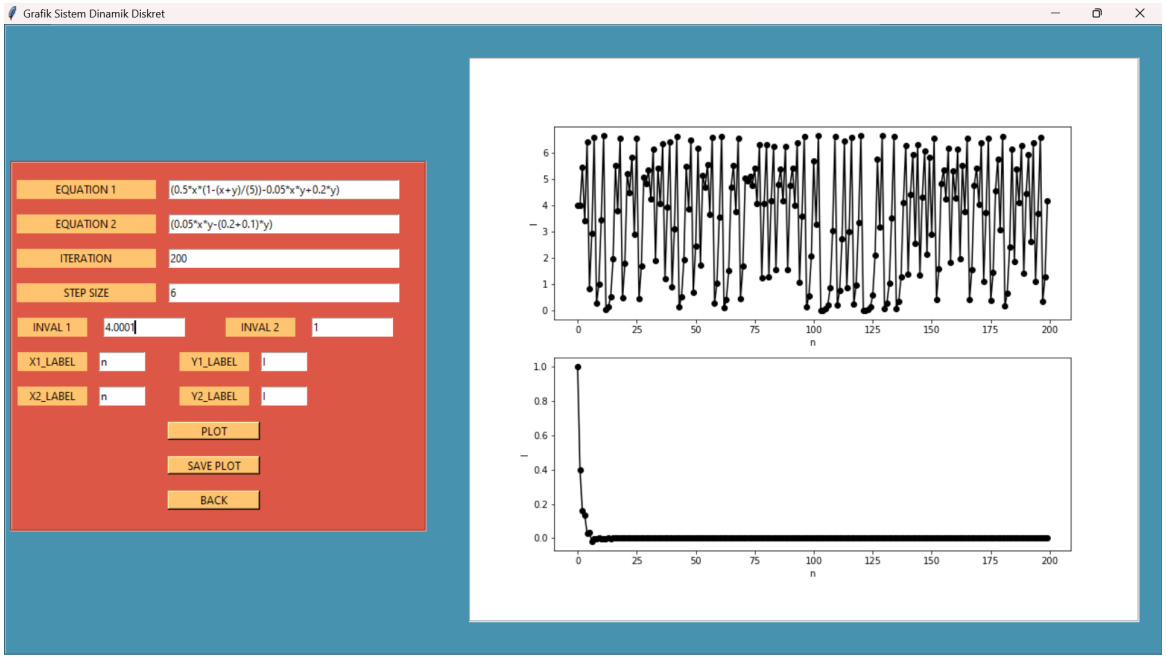


Figure 9. two dimensional time series page

a name to label X1_LABEL and X2_LABEL and Y1 and Y2 LABEL. Press the plot to start plotting the graph. To save the generated graph, press the SAVE PLOT button, and the graph will be saved to the user's device.

Furthermore, the implementation of the Bifurcation graph begins by filling in the EQUATION 1 and EQUATION 2 sections with equations. The equation in this process only involves the variables x and y for other parameters can be written directly, entering the ITERATION value which functions as an iterative calculation process described by the number of points that can be described by the system, entering the TAIL POINT value which functions to provide an overview of the long-term behavior of a

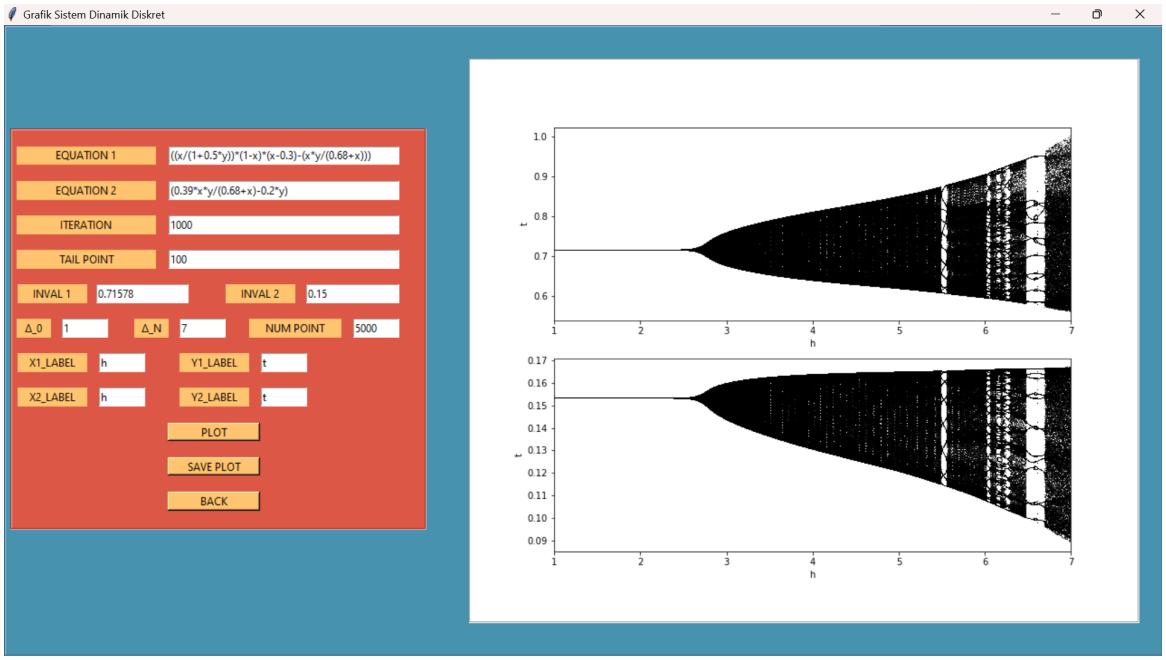


Figure 10. Two dimensional bifurcation page

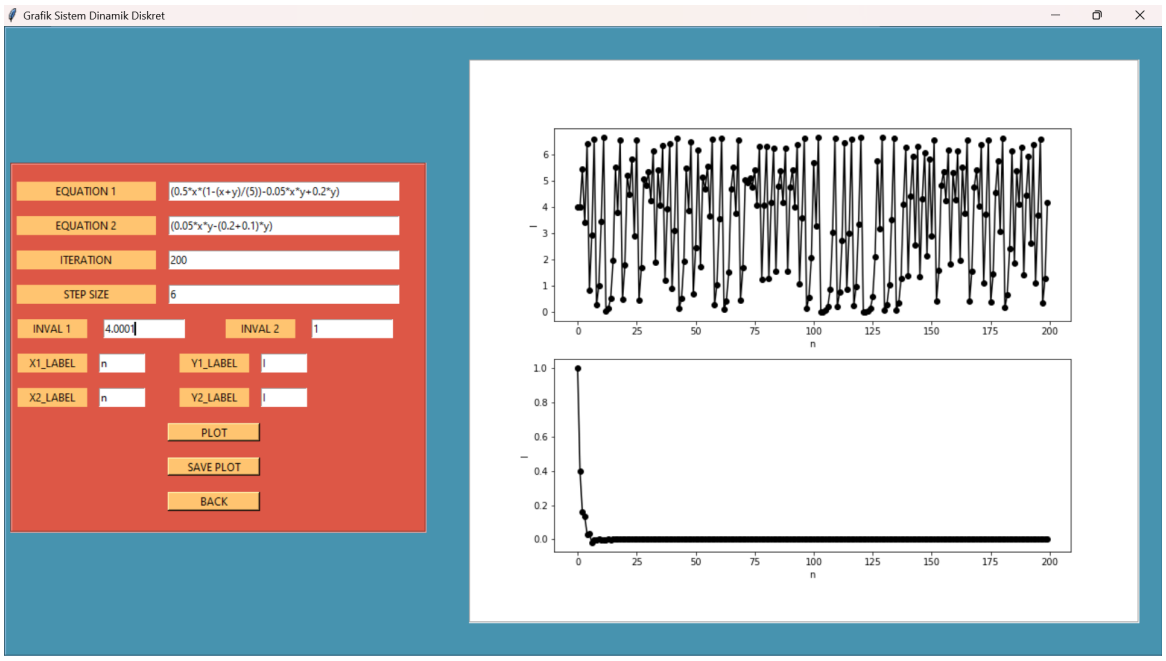


Figure 11. First Validation

system which is described at the end points of the graph, then enter the value of Δ_0 and Δ_N which serves as the initial limit and the final limit of the bifurcation parameters to be described, fill in the value of NUM POINT to control the calculated points in the range, then enter the initial value for INVAL 1 used for EQUATION 1, as well as for INVAL 2 used for EQUATION 2, then enter a name to label the X1_LABEL and X2_LABEL and Y1_LABEL and Y2_LABEL. Press plot to start plotting the graph, if you want to save the graph that has been generated press the SAVE PLOT button, and the graph will be saved on the user's device.

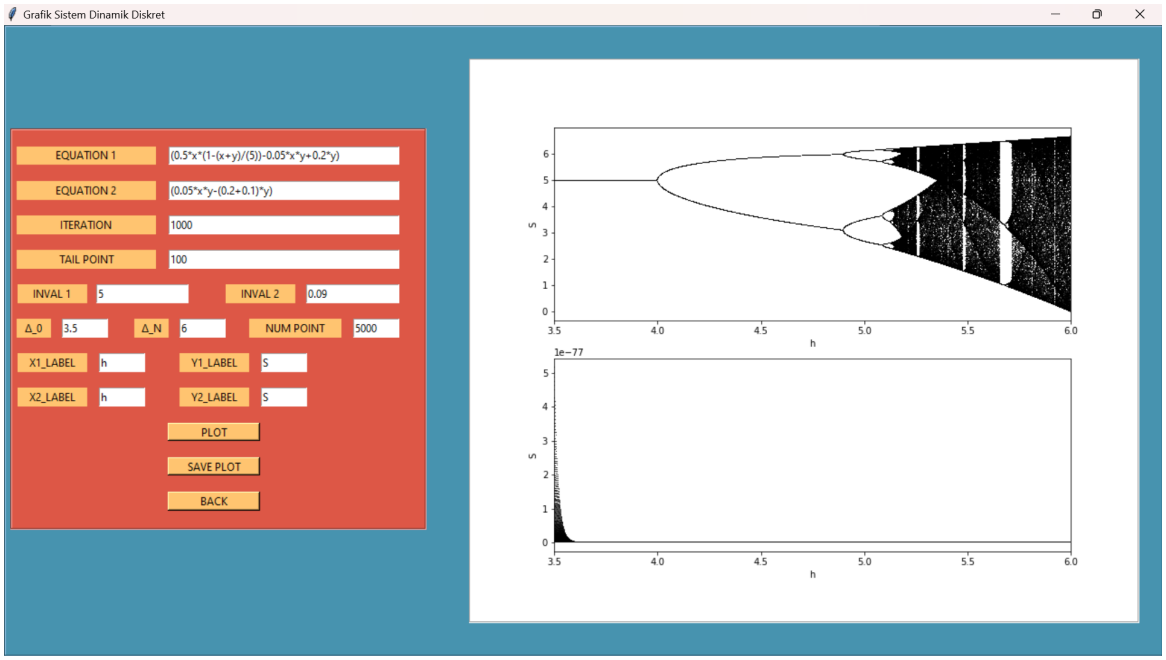


Figure 12. Second validation

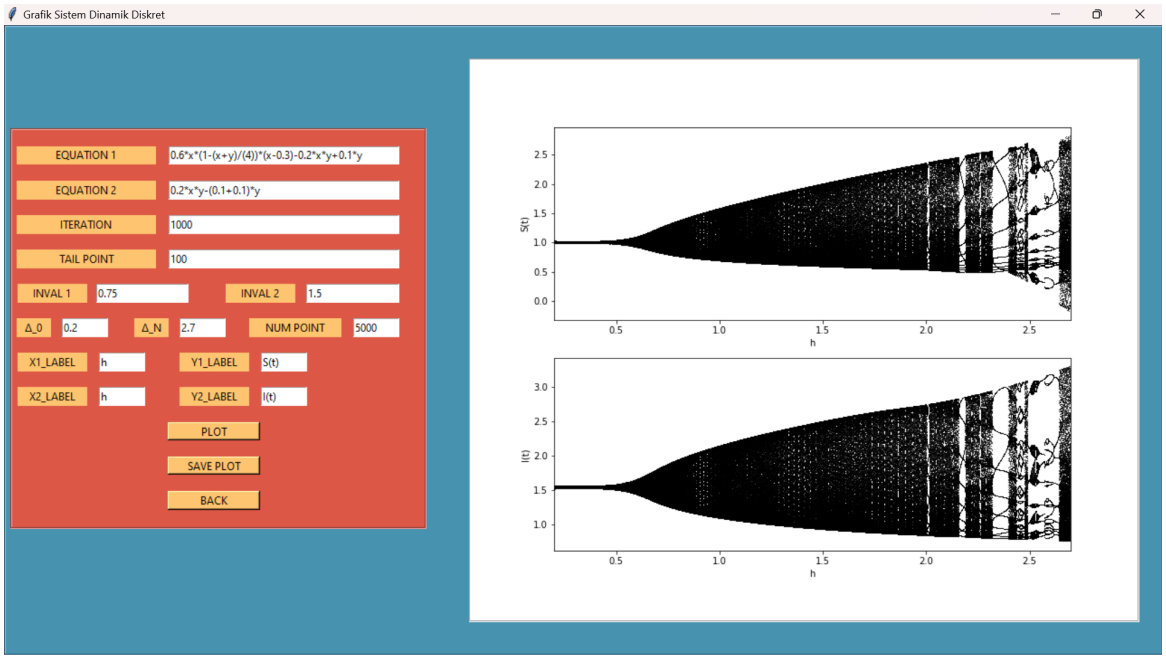


Figure 13. Third Validation

3.1. Verification

At this stage, tests will be conducted on the application to determine whether it has run according to the needs analysis. This application test is conducted by comparing the results of graphs generated using the application with those graphs in the article.

3.1. Maintenance

At this stage, the program can be run according to the commands desired by the user. Further maintenance can be done when there are errors occur when running the program. Maintenance can

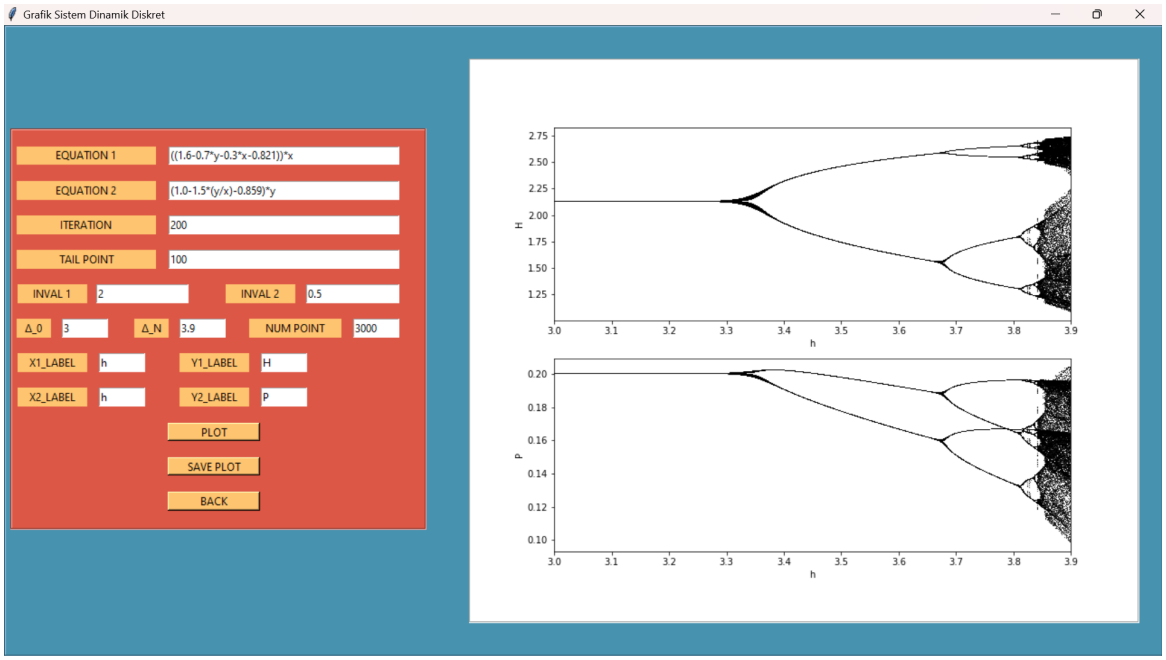


Figure 14. Fourth validation

also be performed if you wish to further develop the program.

3.2. Validation

After applying the waterfall method to create the application, the program will be validated by comparing the graph results in the article with those generated by the application. Validation will be carried out by comparing the graph results in the article [2] using equation 10 below

$$\begin{aligned} S_{n+1} &= \left[rS_n(1-\frac{S_n+I_n}{K}) - \beta S_n I_n + \omega I_n \right] \\ I_{n+1} &= [\beta S_n I_n - (\omega+\delta) I_n] \end{aligned}$$

Then using parameter 16

$$r = 0.5, \quad K = 5, \quad \beta = 0.05, \quad \omega = 0.2$$

From equation 10 and parameter 16, the equation that will be entered in the application is

$$\begin{aligned} x_{n+1} &= \left[rS_n(1-\frac{x_n+y_n}{K}) - \beta x_n y_n + \omega y_n \right] \\ y_{n+1} &= [\beta x_n y_n - (\omega+\delta) y_n] \end{aligned}$$

From the Figure 11, it can be seen that the graph generated in the application is not much different from that in Figure 3 given by Panigoro et al. [2]. For the placement of the starting point and the number of step sizes in the application is in accordance with those in the article. Furthermore, the bifurcation graph using the equation

$$\begin{aligned} x_{n+1} &= [rS_n(1-\frac{x_n+y_n}{K}) - \beta x_n y_n + \omega y_n] \\ y_{n+1} &= [\beta x_n y_n - (\omega+\delta) y_n] \end{aligned}$$

From the Figure 12, it can be seen that the image in the application and Figure 2 part (a) in the article [2] are not much different. For the placement of the starting point, Δ0, and ΔN in the

application is adjusted to produce a similar image in the article. The next validation will compare the graph found in the article [24] using equation 5 and parameter 14. Below is written the equation that will be used

$$\begin{aligned} S &= [rS(1-\frac{S+I}{K})-(S-m)-\beta SI+\omega I] \\ I &= [\beta SI-(\delta+\omega)I] \end{aligned}$$

Because the above equation is still in the form of variables S and I, then S and I will be changed first for S to x and I to y. Then the parameters used are :

$$\beta = 0.2, \quad \omega = 0.1, \quad \delta = 0.1, \quad r = 0.6 \quad m = 0.3, \quad K = 4$$

Based on equation 5 and parameter 14, the equation to be used in the application is written below:

$$\begin{aligned} x &= [0.6*x*(1-\frac{x+y}{4})-(x-0.3)-0.2*x*y+0.1*y] \\ y &= [0.2*x*y-(0.1+0.1)*y] \end{aligned}$$

From Figure 13, it can be seen that the image in the application and the graphic image 1 in the article [24] are not much different. The initial value is in the same position, and for the placement of the starting point, Δ0, and ΔN in the application, adjustments are made to produce a similar image in the article.

The fourth validation will be performed by comparing the graphs presented in the article [25] using Equations 2a and 2b below.

$$\begin{aligned} H_{n+1} &= [(r_1-a_1P_n-b_1H_n-c_1)H_n] \\ P_{n+1} &= [(r_2-a_2\frac{P_n}{H_n}-c_2)P_n] \end{aligned}$$

Using parameters:

$$r_1 = 1.6, \quad r_2 = 1.0, \quad a_1 = 0.7, \quad a_2 = 1.5, \quad b_1 = 0.3, \quad c_1 = 0.821, \quad c_2 = 0.859$$

Based on equations 2a and 2b and the parameter values that have been written, the equations used in the simulation are

$$\begin{aligned} x &= [(1.6-0.7*y-0.3*x-0.821)*x] \\ y &= [(1.0-1.5*\frac{y}{x}-0.859)*y] \end{aligned}$$

From the Figure 14, it can be seen that the image in the application and the image of graph 1 in the article [25] are not much different. The initial value is in the same position and for the placement of the starting point Δ0 and ΔN in the application according to the image in the article.

4. Conclusion

The results of the research show that *Python*-based applications can be run. There are four different programs that can be run on the application: making one-dimensional and two-dimensional Time Series graphs, as well as creating one-dimensional and two-dimensional Bifurcation graphs. Graph generation is performed by manually entering equations and parameters in the application, after which the graph can be saved to the user's device. Validation results are obtained by comparing graphs from the reference article with those from the application, using the same equations and parameters as in the article. The resulting graph can vary according to the equations and parameters selected by the user. The program will experience errors if the equation entered does not use the x and y variables.

Supplementary Information

Acknowledgements. We are incredibly grateful to the editors and reviewers for their helpful comments and invaluable assistance in improving this work.

Funding. This research received no external funding.

Author Contributions. **Farhah Nadhilah:** Conceptualization, Methodology, Writing -Review & Editing, Supervision, Project Administration. **Hasan S. Panigoro:** Conceptualization, Validation, Writing -Review & Editing, Project Administration, Funding Acquisition. **Armayani Arsal:** Supervision, Methodology, Validation, Resources. **Nurwan:** Supervision, Methodology, Validation, Resources. **Djihad Wungguli:** Supervision, Methodology, Validation, Resources. **Isran K. Hasan:** Supervision, Methodology, Validation, Resources. All authors read and approved the final manuscript.

Conflict of interest. The authors declare no conflict of interest.

Data availability. All data presented are hypothetical and cited from prior publications.

References

[1] Cushing JM. Difference equations as models of evolutionary population dynamics. Journal of Biological Dynamics. 2019;13(1):103-27. [Crossref].
[2] Panigoro HS, Rahmi E. The Dynamics of a Discrete Fractional-Order Logistic Growth Model with Infectious Disease. Contemporary Mathematics and Applications (ConMathA). 2021;3(1):1. [Crossref].
[3] Mokodompit R, Nurwan N, Rahmi E. Bifurkasi Periode Ganda dan Neimark-Sacker pada Model Diskret Leslie-Gower dengan Fungsi Respon Ratio-Dependent. Limits: Journal of Mathematics and Its Applications. 2020 jul;17(1):19. [Crossref].
[4] Santra PK, Panigoro HS, Mahapatra GS. Complexity of a Discrete-Time Predator-Prey Model Involving Prey Refuge Proportional to Predator. Jambura Journal of Mathematics. 2022 jan;4(1):50-63. [Crossref].
[5] Roza IZN, Pagalay U, Widayani H. Simulasi Model Diskrit Respon Sistem Imun pada Penyebaran Tumor Otak dengan Metode Beda Hingga Standar. Jurnal Riset Mahasiswa Matematika. 2021 dec;1(2):79-92. [Crossref].
[6] Alya Rahmadani Harahap, Yahfizham Yahfizham. Algoritma Pemrograman Numerik Sebagai Solusi Efisien dari Permasalahan Matematika Kompleks. Jurnal Arjuna : Publikasi Ilmu Pendidikan, Bahasa dan Matematika. 2023;1(6):12-20. [Crossref].
[7] Ra'yan I, Toaha S, Kusuma J. Dynamics Analysis of Predator-Prey Model with Double Allee Effects and Holling Type II Functional Response. Jurnal Matematika Statistika dan Komputasi. 2022;18(3):434-46. [Crossref].
[8] Aini Q, Savitri D. Analisis Dinamik Model Mangsa Pemangsa Holling-Tanner dengan adanya Makanan Tambahan pada Pemangsa. MATHunesa: Jurnal Ilmiah Matematika. 2021 aug;9(2):418-30. [Crossref].
[9] Resmawan R, Yahya L, Pakaya RS, Panigoro HS, Nuha AR. Analisis Dinamik Model Penyebaran COVID-19 dengan Vaksinasi. Jambura Journal of Biomathematics. 2022 jul;3(1):29-38. [Crossref].
[10] Pratama DA, Bakar MA, Fadhilah N. Physical restriction neural networks with restarting strategy for solving mathematical model of thermal heat equation for early diagnose breast cancer. Results in Applied Mathematics. 2023;19:100384. [Crossref].
[11] Guedes PFS, Lacerda MJ, Nepomuceno E. State-Feedback Control Design for Polynomial Discrete-Time Systems obtained via Second-Order Runge-Kutta Discretization. IFAC-PapersOnLine. 2024;58(5):13-9. [Crossref].
[12] Elaydi SN. Discrete Chaos, Second Edition: With Applications in Science and Engineering. vol. 2007. Taylor and Francis; 2007. [Website].
[13] Hanke JE, Wichern DW. Business Forecasting. Pearson Prentice Hall; 2005. [Website].
[14] Hassona S, Marszalek W, Sadecki J. Time series classification and creation of 2D bifurcation diagrams in nonlinear dynamical systems using supervised machine learning methods. Applied Soft Computing. 2021;113:107874. [Crossref].
[15] Qurban M, Khaliq A, Saqib M, Abdeljawad T. Stability , bifurcation , and control : Modeling interaction of the predator-prey system with Alles effect. Ain Shams Engineering Journal. 2024;(September 2023):102631. [Crossref].
[16] Purnomo AB, Prakoso MT, Evi M. Studi kasus ilmu komputer: Efektivitas perancangan model pembelajaran komputasi. Hexatech: Jurnal Ilmiah Teknik. 2022 aug;1(2):55-9. [Crossref].
[17] Suharto A. Fundamental Bahasa Pemrograman Python. Eureka Media Aksara; 2023. [Website].
[18] Dhruv AJ, Patel R, Doshi N. Python : The Most Advanced Programming Language for Computer Science Applications. Proceeding of the International Conference on Culture Heritage, Education, Sustuinable Tourism, and Innovation Technologies. 2022;(Cesit 2020):292-9. [Crossref].
[19] Linge S, Langtangen HP. Programming for Computations – Python. 2nd ed. Barth TJ, Griebel M, Keyes DE, Nieminen RM, Roose D, Schlick T, editors. SpringerOpen; 2018. [Crossref].
[20] Kasliono K, Suharmono E, Povi P, Meriani R, Candraningrum N. Analisis Regresi dan Korelasi untuk Proyeksi Produksi

Minyak Bumi dan Gas Alam Indonesia menggunakan Bahasa Pemrograman Python. Jurnal Teknologi Informatika dan Komputer. 2023 sep;9(2):1297-313. [Crossref].
[21] Moore AD. Python GUI Programming with Tkinter. second edi ed. Jain S, Katare N, Lucy W, Sonawane K, editors. Birmingham: Packt Publishing Ltd.; 2021. [Website].
[22] Hunter J, Dale D, Firing E, Droettboom M. Matplotlib Release 1.5.1; 2016. [Website].
[23] Wahid AA. Analisis Metode Waterfall Untuk Pengembangan Sistem Informasi. Jurnal Ilmu-ilmu Informatika dan Manajemen STMIK. 2020;(October):5. [Website].
[24] Sidik ATR, Panigoro HS, Resmawan R, Rahmi E. The existence of Neimark-Sacker bifurcation on a discrete-time SIS-Epidemic model incorporating logistic growth and allee effect. Jambura Journal of Biomathematics. 2022;3(2):58-62. [Crossref].
[25] Ernawati PD, Darti I. Stability analysis of the euler discretization for the harvesting Leslie-Gower predator-prey model. International Journal of Pure and Applied Mathematics. 2015;105(2):213-21. [Crossref].